Autoencoders

AW

Lecture Overview

1. Building Models of Data

2. Linear Factor Models

Unsupervised Learning

- What happens when we do not have a target variable?
  - We want to capture a model of the training data without the guidance of the target.
  - This is an *unsupervised* learning problem.

Example:

- 2D data set in which all points are distributed on the circumference of an origin-centered circle.
- All points in the first and third quadrant belong to class +1 and remaining points are −1.
  - The class variable provides focus to the learning process of the supervised model.
  - An unsupervised model needs to recognize the circular manifold without being told up front.
  - The unsupervised model can represent the data in only 1 dimension (angular position).
- Best way of modeling is data-set dependent ⇒ Lack of supervision causes problems

Challenges to Unsupervised Learning

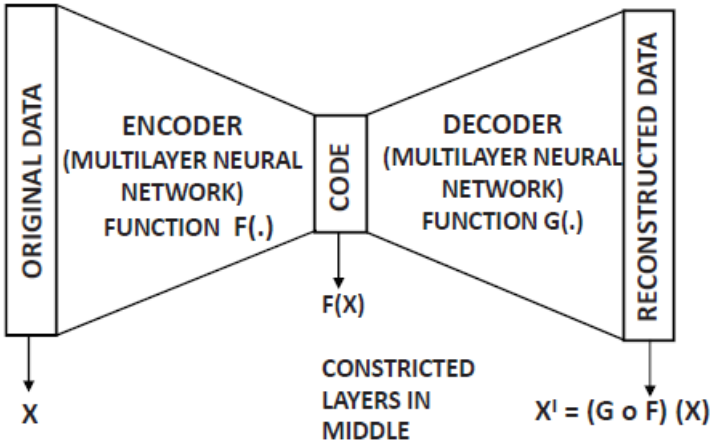
- Cause of the difficulties in unsupervised learning:
  - high dimensionality of random variables being modeled
- The *statistical challenge of* generalization:
  - the number of configurations that must be distinguished grows exponentially with the number of dimensions of interest. It quickly becomes much larger than the number of examples can possibly be used with bounded computational resources.
- The *computational challenge* of intractable computations:
  - Intractable inference involving conditional probabilities distributions of 3 or more vector variables that requires summing up over all possible combinations of values. Clearly this grows exponentially with growth of dimensions
  - Computing normalization constants dependent on partitioning functions

Compression

- Unsupervised models are closely related to compression because compression captures a model of regularities in the data.
- Learning short feature representation implies compression
  - Generative models represent the data in terms of a compressed parameter set.
  - Clustering models represent the data in terms of cluster statistics.
  - Matrix factorization represents data in terms of low-rank approximations (compressed matrices).
- An autoencoder provides a compressed representation of the data.

Encoder and Decoder

- Reconstructing the data might seem like a trivial matter by simply copying the data forward from one layer to another.
- Not possible when the number of units in the middle are *constricted*.
    - Autoencoder is divided into *encoder* and *decoder*
    - Encoder provides compressed representation of data – code that is output $\vec{h}$ encoder hidden layer



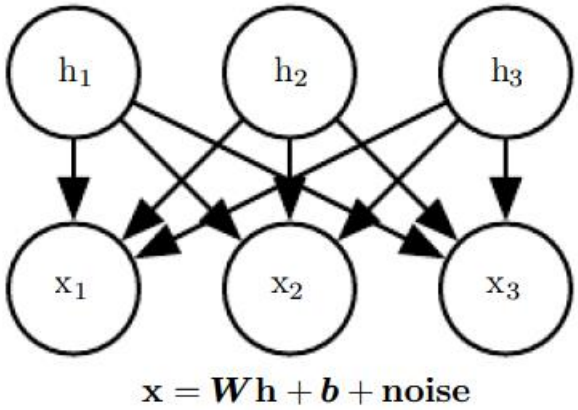
Lecture Overview

1. Building Models of Data

2. Linear Factor Models

Latent Variables and Linear Factor Models

- Most generative models are latent variable models:
  - Construct dependence of target on latent variables that are not directly observed. These variables are inferred from other variables that are directly measured.
  - Goal of autoencoder is to find latent variable representation
- Linear factor models are such that data is generated as follows:
  1. We sample the explanatory factors h from a distribution $\vec{h} \sim p(\vec{h})$ where $p(\vec{h}) = \prod_i p(h_i)$ is a factorial distribution
  2. We sample the real-valued observable variables given the factors: $\vec{x} = W\vec{h} + \vec{b} + \varepsilon \ (= noise)$ where the noise is typically Gaussian and diagonal (independent across dimensions).



Factor Analysis and Probabilistic PCA

Factor analysis:

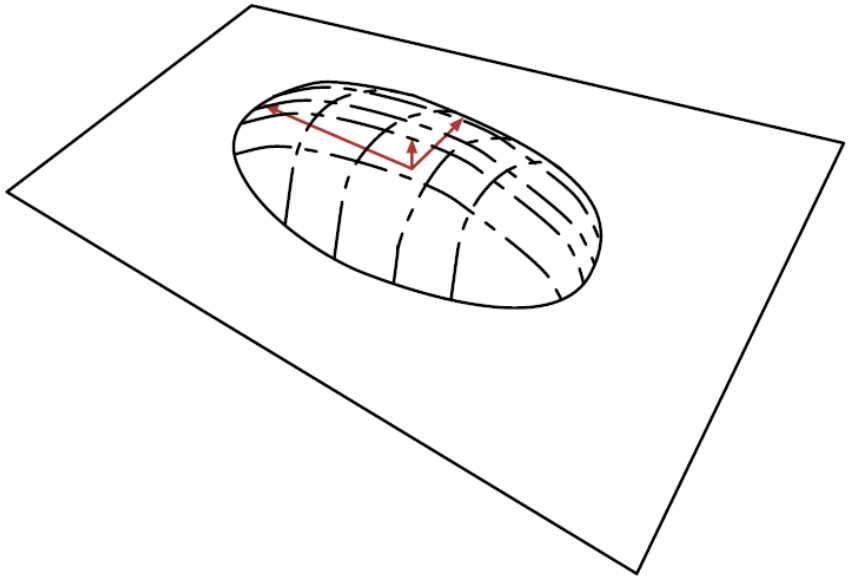
- Latent variable prior is just the unit variance Gaussian $H \sim N(\vec{h};\ \vec{0}, I)$ while the observed variables x_i are assumed to be conditionally independent, given $\vec{h}$
- Noise is drawn from a diagonal covariance Gaussian distribution, with covariance matrix $\psi = \text{diag}(\vec{\sigma}^2)$, with $\vec{\sigma}^2 = [\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2]$ a vector of per-variable variances
- Role of the latent variables is thus to *capture the dependencies* between the different observed variables x_i. It can be shown that $X \sim N(\vec{x};\ \vec{b}, WW^T + \psi)$

Probabilistic PCA:

- Same as factor analysis but σ_i is the same for all i, i.e. $\psi = \sigma I$ so $X \sim N(\vec{x};\ \vec{b}, WW^T + \psi)$ or equivalently $\vec{x} = W\vec{h} + \vec{b} + \sigma\vec{z}$ where $\text{Z} \sim N(\vec{z}; 0,1)$
- There is EM algorithm for estimation of W and σ in PPCA

Manifold Interpretation of PCA

- Probabilistic PCA defines a thin pancake-shaped region of high probability—a Gaussian distribution that is very narrow along some axes, but is elongated along other axes.
- PCA can be interpreted as aligning this pancake with a linear manifold (subspace) in a higher-dimensional space.



Manifold Learning by Linear Autoencoder

- Consider a linear encoder $\vec{h} = f(\vec{x}) = W^T(\vec{x} - \vec{\mu})$. If the goal is to reconstruct $\vec{x}$ as close as possible, i.e. $\hat{\mathrm{x}} = \min_{\vec{\mathrm{x}}_\mathrm{r}} \|\vec{x} - \vec{x}_r\|$ then $\hat{x} = \vec{b} + V\vec{h}$ where $V = (W^T)^{-1}$ and $\vec{b} = \vec{\mu}$, so if W is orthogonal matrix then $V = W$ (as $W^T = W^{-1}$).
- Taking columns of W are orthonormal basis which spans the same subspace as the principal eigenvectors of the covariance matrix $C = (\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T$
- eigenvalue λ_i is the variance of x in the direction of eigenvector v^i. So if $\vec{x} \in \mathbb{R}^D$ and if $\vec{h} \in \mathbb{R}^d$ then the optimal reconstruction error is $\|\vec{x} - \hat{x}\| = \sum_{i=d+1}^{D} \lambda_i$. If $\mathrm{rank}(C) = d$ then error is 0.
- Thus any linear autoencoder that learns matrices W and V with the goal of making the reconstruction of $\vec{x}$ lie as close to $\vec{x}$ as possible learns the linear manifold aligned with axes of $\mathbb{R}^d$

PCA as Feature Selector

When we truncate PCA at fixed number of out of spectrum we reduce dimensionality. But we also select features that we keep this way. Which features are kept then?

- So if $\vec{x} \in \mathbb{R}^D$ and then if $\vec{h} \in \mathbb{R}^d$ then the reconstruction error is $\|\vec{x} - \hat{x}\| = \sum_{i=d+1}^{D} \lambda_{j_i}$. If $\text{rank}(C) > d$ then error is not 0.
- Minimization of L^2 loss requires it to be minimal over all subsets of size $D - d$ determined by complements to images of maps $j: \{1, \dots, d\} \to \{1, \dots, D\}$
- So which features did we keep when we obtained $\vec{h} \in \mathbb{R}^d$?
- Recall that eigenvalue λ_i is the variance of x in the direction of eigenvector v^i. To obtain $\min_i \sum_{j=d+1}^{D} \lambda_{i_j}$ we need to drop dimensions with smallest $D - d$ eigenvalues
- But eigenvalues are variances- so we select features with highest variances
  - Not necessarily most important features!

Reading

- Ch. 2.5.1- 2.5.2 (first part)